POCKET COMPUTER NEWSLETTER

© Copyright 1982 — Issue 16 July

HEWLETT-PACKARD ANNOUNCES PROGRAMMER'S AID

Beginning July 1, 1982, Hewlett-Packard Company will be providing a new "engineering" calculator designed expressly for computer programmers and digital-design engineers. The unit marks a new trend in supporting a highly specialized market by the firm.

The new programmable model has been designated the HP-16C. It is able to handle a wide variety of computer-science problems such as format conversions, bit extraction and *simulation of selected microprocessor instructions!* The register size may be specified by the user, to a maximum of 64 bits, and thus the unit can be set to match the word size of virtually any commercial processor.

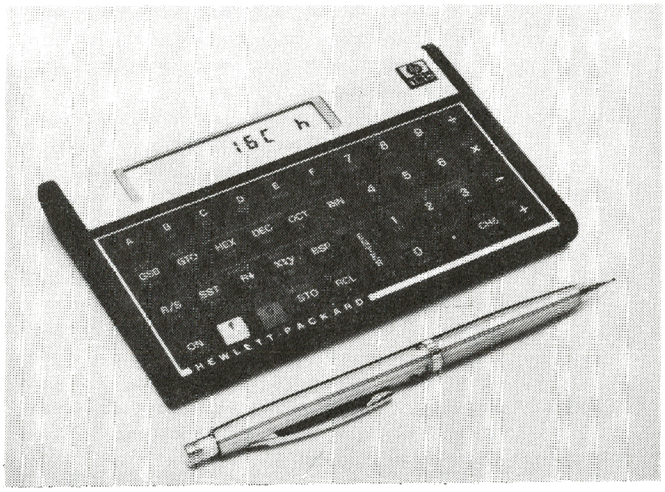
The unit may be placed in an "integer" mode and used to perform bit calculations. In this mode numbers may easily be entered in and converted between four different bases: hexadecimal, decimal, octal and *binary*. It is believed to be the first time that binary notation has ever been made available in a stand-alone handheld calculator. In the unsigned mode, set to a 64-bit word size, numbers up to $2^{64}-1$, which is 18,446,744,073,709,551,615 in base 10, may be represented. The liquid-crystal display scrolls right and left to permit viewing of large numbers.

Users can select whether numbers are to be interpreted as one's-complement, two's-complement or unsigned values. Among other benefits, this capability is said to be a boon to computer science students who sometimes have difficulty learning to distinguish among the three conventions.

The new unit utilizes Reverse Polish notation (RPN) logic and has the same floating-point decimal arithmetic features found in other HP machines. There are also 18 different bit-manipulation functions and four Boolean operators provided as special features in the unit. The programmable portion can retain up to 203 program lines or 101 sixteen-bit data registers.

The suggested U.S. retail list price has been announced as $150.00.

Photo *Hewlett-Packard HP-16C Programmer's Calculator*



UPPER CASE CONVERSION ROUTINE

This routine was submitted by: *David G. Motto, 3639 Roosevelt Circle, Jackson, MI 49203.* It is particularly valuable on the Panasonic/Quasar HHC, but can also be applied on other PCs.

The routine shown in the listing will automatically convert characters in a string from lower case to upper case.

Line 10 accepts the input string (A$) and clears out an area (B$) for the conversion to upper case. Line 20 sets up a loop to extract the characters in the input string, one at a time. Line 30 extracts each character and stores the ASCII value.

Line 40 is the heart of the routine. It forces the ASCII value of the characters to stay in the range 32 to 95. The phrase "C AND 95" provides an upper limit of 95. If the initial value is less than 64, the remainder of the phrase within parenthesis adds 32 to the ASCII value. The resulting modified ASCII value is changed back to a character (now upper case) and is concactenated to the B$ string.

Line 50 completes the scanning loop. Line 60 displays the original input and the converted output.

Some beginning BASIC programmers may not be familiar with the AND function used in line 40. Typically, this logic function is used in IF statements such as IF A=0 AND B=1. However, it may also be used as a numerical operator. This is how it is used in this application.

The number preceding the AND declaration and the number following it are considered to be in binary form (as far as the computer is concerned). Each bit of the two numbers is compared against each other. If both bits are a 1, the result is a 1. Otherwise, the result is 0. Stated logically: if A is true *AND* B is true, then the result is true. If either A or B (or both) are false, then the result is false.

For example:

0	1	1	1	0	0	1	0	represents lower case "r"
0	1	0	1	1	1	1	1	is 95 in binary format
0	1	0	1	0	0	1	0	the result of "r" AND 95

This result is indeed the ASCII representation of upper case R.

So, now you have one more little routine to put in your programming bag of tricks. Note that this routine is particularly valuable when inputting data from the normally lower case HHC as it converts letters of the alphabet, but it does not affect the inputting of numerical digits!

Program *Upper Case Conversion*

```
10  INPUT A$:B$ = ""
20  FOR I=1 TO LEN A$
30  C$=MID$(A$,I,1):C=ASC C$
40  B$=B$+CHR$(C AND 95 + 32 * (C < 64))
50  NEXT I
60  PRINT A$;" ";B$:GOTO 10
```

UNDERSTANDING THE SHARP PC-1500

This is the second article in a series being presented by: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.*

An Introduction to POKE

In Issue 15 of *PCN* I explained that the PEEK function returned the byte that was stored in the address used as an argument. It is also possible to store a specified byte at a particular memory location, by using the POKE instruction.

For example, execution of POKE &79FF,0 will store a zero byte in the memory location &79FF and thus place the PC in the LOCK mode. (Try it!) As you would expect if you read the previous article, performing a POKE &79FF,&60 will cancel the LOCK mode.

A multiple POKE directive is possible on the PC-1500. When the directive POKE &7900,1,0,116,104,80,0,0,0 is executed, the byte 01 is stored into location &7900, 00 goes into &7901, and so on. This particular example will store the numerical value 78.685 into the variable A as discussed in the previous article.

Practical Applications

It may not be immediately obvious that there are a number of ways in which the PEEK and POKE instructions can be quite useful. The following examples should give you some notion of the kinds of possibilities.

For instance, certain memory locations act as *pointers.* That is, they contain the hexadecimal addresses of various memory locations needed by the computer, such as the location of the next instruction to be executed or the location of a stored variable.

Suppose you have executed the NEW command, effectively clearing your program from memory, and then realize you still want to use that program. Is it possible to recover the lost program? It is, provided that prior to having executed NEW you have saved the numbers obtained by PEEK &7867 and PEEK &7868. The contents of these locations act as a pointer to the memory address of the last byte in a program! The use of the NEW command does not actually erase a program from memory. It actually just resets some pointers so that the computer "thinks" that the memory is empty.

To test this, put a small program into the PC-1500 and write down the results of the two PEEKs just mentioned. Now execute NEW. To resurrect your program: (1) POKE the two previously saved PEEKed values back into their original locations and (2) POKE &40C5,0. (Note, if you have an 8K CE-155 memory module installed, step (2) should be POKE &38C5,0.) If the first line number in your program was larger than 255, you will find that it has been changed. You can edit it back to its original value.

You can also restore variables in main memory using a similar procedure. For instance, with some variables stored in main memory (created using two-character names), record the values obtained using PEEK &7899 and PEEK &789A. If these same values are POKEd back into those locations after main-memory variables have been "cleared" by CLEAR, RUN or NEW, the original variables will be restored. You see, the pointer in locations &7899 and &789A simply contains the starting address where variables are stored. (Note that this does not apply to the fixed-memory variables A through Z and A$ through Z$.)

Print formats may be set using the POKE directive instead of a USING statement, with the advantage that a calculated value may be used to determine the number of spaces formatted. Here is how the format settings are stored in memory:

&7895 The byte stored here is the sum of the following format specifications: 128 for scientific notation, 64 for asterisk fill, 32 for forced sign, 16 for comma separation. (Note: when set by USING, the computer adds 1 to these values, except when the format is simply "^".)

&7896 The number of positions reserved preceding the decimal point, including one for the sign.

&7897 Length specified for strings, if any.

&7898 The number of positions reserved following the decimal point, including one for the decimal point itself.

Thus, POKE &7895,32,7,0,3 would be equivalent to the statement USING "+#######.###".

Fooling the Printer

If you have the CE-150 printer connected, you can PEEK &79F4 to see what CSIZE is set. The largest size you can print is CSIZE 9, right? *Wrong!* Try POKE &79F4,50. Then LPRINT "G" and take a look at CSIZE 50! Now set the computer to GRAPH mode, execute the statement ROTATE 1, then POKE &79F4,36. Now execute LPRINT "HUGE". Be advised that if you continue this for very long, the PC will exceed its coordinate range resulting in ERROR 70. You can remedy this condition by executing SORGN so that it has a new origin. You might also discover that the computer interprets CSIZE 0 as CSIZE 256. The only thing printable in this size is an immense decimal point, obtainable by specifying LPRINT ".".

The COLOR setting is stored in location &79F3. Do *not* POKE into this location. You will only confuse the computer into thinking that the pen turrett is positioned differently from where it actually resides.

[Note: Be careful using the POKE directive around memory locations used to control external units. It is conceivable that you might damage an external device if you inadvertently left a selenoid activated or otherwise interfered with the normal programmed operation of the device. At the very least, be prepared to immediately shut the unit off if strange sounds or activities come forth as the result of a POKE directive. – N.W.]

More PEEK and POKE Locations

I have located a number of memory addresses used by the PC-1500 using a process of trial and error. And, of course, there are quite a few addresses used for purposes that I have not yet figured out. The accompanying table lists those found recently. Be forewarned that if you poke values into certain locations that attempt to have the computer do something that is not possible, you may get a "crash." The PC may then completely lock up and you will have to use the ALL RESET switch to restore operation.

A Surprise PEEK

A number of PC-1500 fans are investigating the ROM coding, obtaining the hexadecimal codes and/or interpretations of these as ASCII characters. Here is another way of looking at the ROM that you may not have considered. The program illustrated can be used to display ROM codes as graphics on the LCD.

```
10 INPUT "BEGINNING ADDRESS?";A
20 WAIT 0:FOR C=0 TO 127:GPRINT PEEK(A+C);:NEXT C:WAIT:
   PRINT
30 A=A+128:GOTO 20
```

Try using &C000 as the initial input to the beginning address prompt. When you get bored with this, take a look at the results when you use &FCA0 as a beginning address!

POINTERS

&7863	Beginning of memory. This location contains the first byte of the two-byte adress of where memory begins. Only one byte is stored because the second byte is always zero. The beginning of memory can vary depending on what expansion module, if any, is plugged into the PC.
&7864	Top of memory. Again, the second byte is always zero. This is the address of the first byte following the end of "main" (user) memory.
&7865 – 66	Beginning of program. This is the address following RESERVE memory. It is where the first program line of a user program is stored.
&7867 – 68	End of program. The computer stores a byte with the code &FF in the address pointed to by this pointer. Thus &FF is the byte immediately after the carriage-return (ENTER) code of the user's last program line.
&7869 – 6A	Beginning of merged program (a program loaded from cassette using the MERGE command).
&7899 – 9A	Beginning of variables.

POINTERS USED DURING PROGRAM EXECUTION

&789C – 9D Hexadecimal value of line number being executed.

&789E – 9F Beginning of the program being executed. (It is the start of a merged program if that is what is being executed.)

&78BE – BF Pointer to address at which next READ statement will obtain data. If none, the value C0C8 is held in the pointer.

SIX-BYTE POINTERS USED IN PROGRAM EXECUTION

Each of these pointers, starting at the location shown, contains six bytes. The first two bytes specify the storage address in memory; the next two contain the program line number; and the last two hold the memory address of the first byte of the program being executed.

&78A0 Address of next program byte. It is the next byte to be executed unless a transfer (GOTO, etc.) has been specified.

&78A6 Address of next byte to be executed after a transfer.

&78AC Address where execution is to resume after a halt for input, etc.

&78B2 Address of program location at which an error occurred.

&78B8 Transfer destination specified by ON ERROR GOTO directive. When cancelled by RUN, &80 is added to the first byte of this stored address.

STORED MODES AND SETTINGS

&764E Status of prefix keys; specification of RESERVE group.

&764F Operating mode and trigonometric mode.

&7863 Beeper on/off. Set to 0 for on, 1 for off.

&7871 – 73 WAIT specification.

&7874 – 75 Position of display cursor (GCURSOR).

&788D Trace mode. TRON = 96, TROFF = 0.

&7895 – 98 Format (USING) specifications. (See text.)

&79FF Lock mode. LOCK = 0, UNLOCK = 60.

&7B0A – 0C Automatic 7-minute shutoff timer. Reset to value &FE 1D 1D whenever activity halts. Shuts PC off when value &FF FF FF is passed.

&7B0D Timer for cursor flashing.

&7B0E Type of program execution in progress. Continuous execution = 1, single-line execution (stepped by user with the line scroll-down key) = 193, and rapid execution when scroll-down key is held depressed = 225.

&7B0F Keyboard matrix code for the key that initiated program execution. This would be ENTER or user-defined or a RESERVE key.

LOCATIONS RELATED TO PRINTER OPERATION

&79E0 – E1 The X-coordinate relative to the origin when in the GRAPH mode. Negative numbers are stored in complemented form.

&79E2 – E3 The Y-coordinate.

&79E4 – E5 Paper reverse counter. Used to insure that paper does not get backed up more than about 4 inches.

&79E6 Location (horizontal) of pen (0 – 216).

&79E7 – E8 In GRAPH mode, the amount by which the horizontal coordinate exceeds the range in which printing may take place. This is in hexadecimal notation with the low byte first.

&79E9 When paper is fed by the computer the value 15 is stored here. When fed manually, the value is 0.

&79F2 ROTATE setting.

&79F3 COLOR setting.

&79F4 CSIZE setting.

CONVERTING PROGRAMS FOR THE CASIO FX-702P

Several *PCN* readers have expressed dismay over the lack of programs available for the Casio FX-702P. A recent survey of PC users leads us to believe that the situation is not likely to improve substantially in the immediate future. There are simply relatively few FX-702P users compared to Radio Shack and Sharp owners, according to our findings.

But, FX-702P owners need not despair. The fact is that most of the programs published to date in *PCN* are readily adapted for use on the Casio PC by making some editing changes as programs are keyed in.

The following suggestions and tips may be helpful to Casio FX-702P users who wish to make such program adaptations.

1.) Substitute appropriate statement/function mnemonics where applicable. Most of these types of alterations are pretty obvious. Thus, for instance, GOSUB is changed to GSB, PRINT to PRT, RETURN to RET, and so forth. A few may not be quite so apparent. The PAUSE statement becomes WAIT (x) where (x) specifies the length of the pause. CLEAR must be changed to VAC on the 702P. You need to type in SQR in place of the square root ($\sqrt{}$) symbol. Also, remember that there is a special inequality symbol ($\neq$) on the 702P that takes the place of having to use the "less than" and "greater than" (< >) symbols to express the inequality operation on a Sharp or Radio Shack PC.

2.) Forget about functions that do not exist on the FX-702P – such as BEEP.

3.) Shorten/abbreviate display messages to fit in the 20 character FX-702P display (versus the 24 characters on the PC-1).

4.) If subscripted variables are used in a program, remember the following:

A.) Subscripted variables having subscripts in the range 1 – 26 are really just another way of declaring the use of the variables A – Z on the Sharp/Radio Shack original PCs. Thus A(1) is the equivalent of specifying variable A on a FX-702P, A(2) = B,.... ..., A(26) = Z. Make the appropriate substitutions!

B.) Subscripted variables with subscripts above 26 can be converted to array elements on a FX-702P using the formula (I-26) where "I" represents the subscript on the Radio Shack/Sharp unit. I.e., A(27) = A(27-26) = A(1), A(28) = A(28-26) = A(2) on the 702P, etc. Simple, isn't it?

5.) Use separate IF statements in place of multiple logic expressions. An expression such as "IF (A < 52) + (B > 60) THEN 400" on a Radio Shack TRS-80 PC means "if A less than 52 *OR* B greater than 60 then go to line 400." On the FX-702P this must be precisely defined using two separate (consecutive) IF... THEN statements, such as:

```
IF A < 52 THEN 400
IF A > 60 THEN 400
```

The opposite logic operation (*AND*) would appear in a TRS-80 listing as "IF (A < 52) * (B > 60) THEN 400" which indicates that "if A is less than 52 *AND* B is greater than 60 then go to line 400." Such a statement can be re-arranged so that it can be stated using two consecutive IF statements on a 702P in this manner:

```
IF A >= 52 THEN ***      (*** = condition not met)
IF B > 60 THEN 400
*** (line to which program goes when condition is not met!)
```

Note that the first IF statement in this example reverses the original test so that the program can "fall" directly into the second IF statement if the first part of the AND operation is met.

6.) Explicitly state all mathematical operations using their appropriate symbols. This means eliminating all instances of "implied" multiplication and parenthesis which sometimes are inferred in TRS-80 and Sharp PC programs. That is, an expression such as:

```
ATN((ABC + (D - E
```

should be completely specified by inserting appropriate multiplication signs and adding closing parentheses, thus:

```
ATN((A*B*C) + (D - E))
```

Use the six steps outlines in this article as a guide and you will find you can readily adapt most PC programs written for other machines so that they will execute on your FX-702P. In the process, you may even find you *understand* (and can thus adapt and customize) the program better than those who simply load the programs "to see what happens."

A COMPARISON OF PC MEMORIES

The Radio Shack TRS-80 and the Sharp PC-1211 pocket computers have 204 registers available for the storage of data and programs. Each register can hold eight bytes. The first 26 registers can only hold *data* (not program steps). Registers can hold numbers of up to 12 digits, a two-digit exponent (powers of ten) and the signs of the mantissa and exponent. Each digit or sign takes up 4 bits of space or a *nibble.* Alternately, a register may store up to seven alphanumeric characters. Each characters is represented as 8 bits or a *byte.* The eighth byte in the register is used to indicate that the other bytes are representing *string* (alphanumeric) information.

The remaining 178 registers can hold data or program steps. Program steps begin at the last register (number 204) and proceed to be stored in descending (numbered) registers. Each program step uses one byte of storage. The *curtain* between program memory and data memory is moved automatically as program steps are created or deleted.

Registers may be cleared in two ways, depending on which type of register is involved. The NEW command effectively clears all program and data registers. The CLEAR command only affects the data registers.

If a data register contains a numerical value, such as 123, then trying to refer to that register as a string will result in an error condition.

A Look at the Casio FX-702P

This unit has 236 data and program registers. Twenty-six of these are reserved strictly for data purposes and 10 are reserved for program steps. The remaining 200 may be used for either data or program storage.

The Casio uses the same method of storage as the TRS-80 and 1211. The curtain, however, must be specifically specified by using the DEFM command. This directive allocates up to 200 registers (by groups of 10) for data storage. It will not permit allocation of memory that is already in use for programs. But, when the curtain is moved to create more data storage, that memory is cleared.

The VAC command clears data memory in the Casio unit and the CLR command clears program memory.

If a data register contains a number and an attempt is made to refer to it as a string, there is no error condition. Instead, the computer returns the "null" string (as though there was nothing there).

The Sharp PC-1500 & Radio Shack PC-2

These units are different than the earlier PC-1211 and PC-1. They have 26 fixed numeric registers, 26 separate fixed string registers, and 1850 bytes of program storage. The numeric data registers behave the same as described for the other units. However, the string registers are twice as long, holding 16 characters.

The 1850 bytes of storage memory may also be used for data when needed. To do so, a two-character variable name is created . The first character of the name may be any letter from A to Z. The second character may be a letter or a digit. Adding the $ symbol creates a string variable.

Arrays of one or two dimensions may also be created in memory through use of the DIM statement. Both numerical arrays and string arrays having up to 80 characters-per-element may be defined. The assignment of an array name takes seven bytes of memory. Additional memory is then used by each element of the array.

The CLEAR and NEW commands operate similar to the earlier PCs by these firms. However, variables (including arrays) in the program workspace are not set to zero by the CLEAR directive; that memory space simply becomes available for other use.

The RUN command, used to start a program, clears all variables in the program workspace, but it does not alter those in fixed memory.

The Panasonic HHC

The Panasonic does things differently than the other machines. It does not have any fixed registers. Each variable, including arrays, uses up program workspace as it is needed. There are three types of variables and array elements: floating-point numbers, integers and strings. Floating-point numbers have up to 9 digits of accuracy and may range from 2.93873588E-39 to 1.70141183E+38 (plus or minus). Integer values are restricted to the range -32768 to 32767. Strings may contain up to 255 characters.

Each simple variable name (versus array variables) uses up seven bytes of memory. Floating-point variables utilize the first two bytes for the name and the last five for the numeric value. Integers use the first two for the name, the next two for the value, and leave the last three unused!

String variables use two bytes for the name, one for the length of the string, and two more for the memory address of the location of the first character of the string. The last two (of the seven) bytes are not used in a string assignment.

Array names use two bytes for the name, two for the array length, one for the number of dimensions and additional bytes for the size of each dimension. Then the storage of array elements begins. Floating-point elements require five bytes, integer elements two bytes, and string elements use three bytes in the array (length and location) with the actual strings being stored separately.

Summary

Each PC or HHC handles certain aspects of memory utilization in its own way. However, there is a clear trend towards providing substantially more memory, both in the basic unit and as add-on or plug-in modules. For instance, the PC-1500 and PC-2 can be expanded using plug-ins to (eventually) 18 kilobytes. The Panasonic, using external modules, can be equipped with up to 56 kilobytes!

An accompanying table compares the basic memory configurations of the units discussed.

Thanks for this comparative article go to: *David G. Motto, 3639 Roosevelt Circle, Jackson, MI 49203.*

Table *Basic Memory Supplied in Popular Units*

Model	Fixed Data	Flexible Memory	Fixed Program
1211/PC-1	208	1424	0
FX-702P	208	1600	80
1500/PC-2	624	1850	0
RL-H1400	0	3108	0

Glossary *PC Storage Terminology*

bit	The smallest unit of programmable memory, able to represent a zero or a one.
nibble	A group of four bits, able to hold a number from 0 to 15.
byte	Two nibbles or eight bits, capable of representing numbers in the range 0 to 255.
BCD	Binary Coded Decimal, a method of representing the digits 0 through 9 in four bits.
curtain	Also known as a partition. It is the hypothetical boundary between program memory and data memory.
fixed memory	Memory that is available for only one purpose, be it storage of data or programs.
flexible memory	Memory that may be used for either data or programs.
program memory	Memory in which program steps are stored.
data memory	Memory in which numbers or character strings are stored.

MUSIC ON THE SHARP PC-1500

Incredible! J.S. Bach's *Prelude Number II from The Well Tempered Clavier* as programmed by: *Brian Peterson, 6807 N. Sheridan Road – Apt. 520, Chicago, IL 60626.* Brian says he will write a tutorial article showing the rest of us how he makes this kind of music if there is enough interest. So, if that is the case, write and give him some encouragement!

Program *Music on the Sharp PC-1500*

```
10:DIM A(51),B(51)
15:FOR X=1TO 51
20:READ A(X):B(X)=10^(X/40)*8
25:NEXT X
30:WAIT 25:PRINT "ONE":PRINT "AND":
   PRINT "TWO":PRINT "AND":CLS
50:READ N:IF NBEEP 1,A(N),B(N)*1.2:
   GOTO 50
51:BEEP 1,A(11),B(11)*4
52:READ N:IF NBEEP 1,A(N),B(N)*2:
   GOTO 52
53:READ N:IF NBEEP 1,A(N),B(N)*1.5:
   GOTO 53
54:BEEP 1,A(11),B(11)*4
55:READ N:IF NBEEP 1,A(N),B(N)*2:
   GOTO 55
56:READ N:IF NBEEP 1,A(N),B(N)*1.5:
   GOTO 56
57:BEEP 1,A(11),B(11)*4
58:READ N:IF NBEEP 1,A(N),B(N)*2:
   GOTO 58
59:READ N:IF NBEEP 1,A(N),B(N)*1.5:
   GOTO 59
60:READ N:IF NBEEP 1,A(N),B(N):GOTO
   60
61:BEEP 1,A(16),B(16):BEEP 1,A(23),
   B(23):BEEP 1,A(26),B(26):BEEP 1,
   A(28),B(28)
62:BEEP 1,A(32),B(32)*28
63:READ N:IF NBEEP 1,A(N),B(N)*2:
   GOTO 63
64:READ N:IF NBEEP 1,A(N),B(N)*1:
   GOTO 64
65:READ N:IF NBEEP 1,A(N),B(N)*4:
   GOTO 65
66:READ N:IF NBEEP 1,A(N),B(N)*2:
   GOTO 66
67:BEEP 1,A(16),B(16):BEEP 1,A(21),
   B(21):BEEP 1,A(24),B(24):BEEP 1,
   A(28),B(28)
68:BEEP 1,A(33),B(33)*28
69:READ N:IF NBEEP 1,A(N),B(N)*2:
   GOTO 69
70:READ N:IF NBEEP 1,A(N),B(N)*1:
   GOTO 70
71:READ N:IF NBEEP 1,A(N),B(N)*1.2:
   GOTO 71
72:READ N:IF NBEEP 1,A(N),B(N)*2:
   GOTO 72
73:READ N:IF NBEEP 1,A(N),B(N)*4:
   GOTO 73
74:READ N:IF NBEEP 1,A(N),B(N)*6:
   GOTO 74
75:READ N:IF NBEEP 1,A(N),B(N)*8:
   GOTO 75
76:BEEP 1,A(32),B(32)*2:BEEP 1,A(30
   ),B(30)*2:BEEP 1,A(32),B(32)*30
77:END
100:DATA 255,244,229,218,204,195,184
   ,172,163,151
101:DATA 143,133,125,118,111,105,99,
   94,88,82,77
102:DATA 72,68,64,59,55,52,49,45,43,
   40,37,35
103:DATA 32,30,28,26,24,22,21,19,18,
   16,15,14
104:DATA 12,11,10,9,8,7
200:DATA 40,31,30,31,28,31,30,31
201:DATA 40,31,30,31,28,31,30,31
202:DATA 36,33,32,33,28,33,32,33
203:DATA 36,33,32,33,28,33,32,33
204:DATA 39,33,31,33,30,33,31,33
205:DATA 39,33,31,33,30,33,31,33
206:DATA 40,35,33,35,31,35,33,35
207:DATA 40,35,33,35,31,35,33,35
208:DATA 43,36,35,36,31,36,35,36
209:DATA 43,36,35,36,31,36,35,36
210:DATA 42,34,32,34,30,34,32,34
211:DATA 42,34,32,34,30,34,32,34
212:DATA 42,35,34,35,30,35,34,35
213:DATA 42,35,34,35,30,35,34,35
214:DATA 40,32,30,32,28,32,30,32
215:DATA 40,32,30,32,28,32,30,32
216:DATA 40,33,32,33,28,33,32,33
217:DATA 40,33,32,33,28,33,32,33
218:DATA 38,33,31,33,30,33,31,33
219:DATA 38,33,31,33,30,33,31,33
220:DATA 38,35,33,35,31,35,33,35
221:DATA 38,35,33,35,31,35,33,35
222:DATA 36,35,33,35,31,35,33,35
223:DATA 36,35,33,35,31,35,33,35
224:DATA 36,30,28,30,26,30,28,30
225:DATA 36,30,28,30,26,30,28,30
226:DATA 35,26,24,26,31,26,24,26
227:DATA 35,26,24,26,31,26,24,26
228:DATA 33,28,26,28,25,28,26,28
229:DATA 33,28,26,28,25,28,26,28
230:DATA 33,30,28,30,27,30,28,30
231:DATA 33,30,28,30,27,30,28,30
232:DATA 33,30,28,30,27,30,28,30
233:DATA 33,30,28,30,27,30,28,30
234:DATA 31,28,27,28,23,28,27,28
235:DATA 31,28,27,28,23,28,27,28
236:DATA 21,31,30,31,33,31,30,31
237:DATA 21,31,30,31,33,31,30,31
238:DATA 22,28,27,28,31,28,27,28
239:DATA 22,28,27,28,31,28,27,28
240:DATA 31,28,27,28,23,28,27,28
241:DATA 31,28,27,28,23,28,27,28
242:DATA 34,28,27,28,25,28,27,28
243:DATA 34,28,27,28,25,28,27,28
244:DATA 35,28,27,28,30,28,27,28
245:DATA 35,28,27,28,30,28,27,28
246:DATA 36,28,27,28,30,28,27,28
247:DATA 36,28,27,28,30,28,27,28
248:DATA 0,15,18,21,0,24,21,20,21,27
   ,21,30,27,24,21,20,21
249:DATA 0,16,19,23,0,28,23,22,23,31
   ,28,35,31,28,24,23,24
250:DATA 0,13,22,28,0,31,28,27,28,34
   ,28,37,34,31,28,27,28
251:DATA 11,0,42,40,42,43,40,39,40,3
   7,40,39,40,42,39,37,39
252:DATA 35,39,37,39,40,37,35,37,34,
   37,35,37,39,35,34,35
253:DATA 38,47,45,47,48,45,43,45,42,
   45,43,45,47,43,42,43
254:DATA 40,43,42,43,45,42,40,42,39,
   42,40,42,43,40,39,40
255:DATA 35,40,39,40,36,45,43,45,35,
   43,42,43,33,42,40,42
256:DATA 31,40,39,40,36,33,31,33,35,
   31,38,31,33,30,28,30,0
257:DATA 28,30,32,0,33,35,36,38,40,3
   8,36,35,0,33,0,35,32,0
258:DATA 35,33,32,33,35,36,35,0,33,3
   1,30,31,33,30,31,33,0
259:DATA 27,4,15,18,21,24,23,21,27,2
   1,30,21,27,24,23,21
260:DATA 20,29,26,23,28,24,21,24,23,
   26,23,20,24,21,18,21
261:DATA 20,23,20,16,21,18,15,18,0,4
   ,11,16,18,20,23,26,23
262:DATA 24,28,33,30,33,36,0,40,39,4
   0,0,35,33,0,30,0,0
263:DATA 0
```

STATUS 1

3403

STATUS REPORT

A number of PC-1500 users have discovered that in addition to the STATUS 0 and STATUS 1 functions available on the machine, there are several other STATUS-related capabilities. Here is a summary of STATUS functions now reported to be available:

STATUS 0 As indicated in the PC-1500 Instruction Manual, this function tells how many bytes of program memory are available for storage of programs and data. This function does not take account of memory assigned to two-character variables or array elements.

STATUS 1 As reported in the manual, this function tells how many bytes of memory a user's program *listing* has consumed. It does not include the space necessary for the storage of two-character variables or array elements.

STATUS 2 This function is not covered in the manual, but it apparently gives the actual address in memory where a user's program listing ends. If you enter

STATUS 2 - STATUS 1

you will obtain the memory address where user program storage actually begins. This address varies depending on which (if any) memory expansion modules are in use.

STATUS 3 Also not reported in the manual, this function returns the "bottom" of the two-character variables and/or array elements stack. Thus, to find out how much memory you have left when you want to include assignments for two-character variables and array elements, just enter:

STATUS 3 - STATUS 2

STATUS 4 Again is not reported in the manual, but it gives the last *complete* line number executed in a program. You can thus use this function to perform a lot of programming tricks – such as determining what line caused a jump because of an ON ERROR GOTO directive. You can also use this function to determine what part of a large program called a subroutine, etc. You just insert:

X = STATUS 4

in the first line of the subroutine and use X as desired to accomplish your programming objectives.

STATUS 5–9 Appear to duplicate the STATUS 4 function.

Thanks to readers such as *Thomas S. Cox, Brian Peterson, Norlin Rober* and others for reporting these latent capabilities.

Error Codes *Summary courtesy of Thomas S. Cox*

PC-1500 ERROR CODES

CODE	MEANING
1	Syntax Error Ex: 10 GOTO: 10 5A=1: a=6: 10 NEW
2	No FOR to match NEXT or no GOSUB to match RETURN
4	Out of DATA
5	Double Diminension--Array already Defined
6	Trying to use an Array Variable before defining it
7	Improper Variable type (String or Numeric)
8	Attempt to Dimension array beyond 2
9	Subscript Range Error. Subscript exceeds Dimension.
10	Out of Memory. Not enough memory to create a new variable.
11	Undefined Line Number
12	Incorrect Format in a USING command
13	Out of Memory in Program or Reserve Area
14	FOR/NEXT nested too deeply or buffer space exceeded
15	GOSUB/RETURN nested too deeply or string buffer space exceeded
16	OVERFLOW/UNDERFLOW or HEX exceeds 65535 decimal
17	Mixed String and Numeric data in mathematical expression
18	Inapprorpriate arguments such as MID$("jkl") or COS (45,80)
19	Dimension exceeds 255
20	When using Fixed Memory Array Variable (@), parenthesis not used
21	Variable required in expression
22	No memory available when program is loaded
23	Improper number entered for setting TIME
26	Command is not executable in current MODE
27	No program corresponding to given label: no printer
28	Command inserted inside " " or improper INPUT or AREAD
30	Line number exceeds 65539: Error 1 if line number 65280-65539
32	Graphic Cursor between coloumns 152-155 when executing INPUT
36	Data can't be specified with format given
37	Numeric calculation overflow
38	Division by zero
39	Negative Log or Negative SQR or improper trig. argument
CASSETTE RELATED ERRORS:	
40	Inappropriate Specification for expression
41	SAVE or LOAD specified for ROM area
42	Cassette file too large, can't be CLOADED
43	CLOAD? data format does not match file format
44	CHECKSUM error
PRINTER RELATED ERRORS:	
70	Pen has exceeded -2048<=X,Y<=2047 or will exeed if executed
71	Paper has or will back up more than 10.24 cm in TEXT mode
72	Inappropriate value for TAB
73	Wrong printer mode (GRAPH or TEXT)
74	Too many commas in LINE or RLINE command
76	Calculated result can't be LPRINTED on one line in TEXT mode
78	Pens being changed or LOW BATTERY state not corrected
79	COLOR signal has not been received
80	Low battery

UPDATES TO THE PC-1500 INSTRUCTION MANUAL

Add the following to page 109:

3. The CLOAD command

When using the CLOAD? command it is necessary that the tape be cued to the carrier tone which precedes the recorded data. The carrier tone consists of a steady, unmodulated signal of approximately 1 kHz. If the tape is not properly cued, no error will be indicated and no data will be presented on the display.

Replace paragraph 5 on page 111-112 with the following:

5. Use of the MERGE command

The MERGE command allows you to store many programs in the computer memory at the same time. For example, let's assume the computer memory contains the following program:

```
10: PRINT "DEPRECIATION ALLOWANCE"
20: INPUT "Enter method:  ";A
```

At this point you remember that you have a similar program portion on tape under the filename "DEP1". You will, of course, want to see if this program has sections useful in the program you are currently constructing. This first step is to find the tape with "DEP1" on it. Cue the tape to the place at which "DEP1" starts. Now type MERGE "DEP1" and press ENTER. The computer will now load "DEP1" into memory *in addition* to the above program. After "DEP1" is loaded, you might find something in memory similar to this:

```
10: PRINT "DEPRECIATION ALLOWANCE"
20: INPUT "Enter method:  ";A
10: "DEP1":REM - - Second Module - -
20: PRINT "INTEREST CHARGES"
30: INPUT "Amount Borrowed: ";B
  :
(etc.)
```

Note that unlike the CLOAD command, the new program *did not* replace the existing one and that some line numbers have been duplicated. Also note that a "label" was used on the first line of the merged module. This allows *linking* of the modules together (see Linking Merged Modules – below).

It is important that you review the following information before proceeding with any further editing or programming:

Important Notes

Once a MERGE is performed, *no insertions, deletions or changes are allowed to previously existing program lines.* Example:

```
10  "A"  REM  This is existing program
20  FOR T=1 TO 100
30  LPRINT T
40  NEXT T
    (etc.)
```

Before doing a MERGE of the next program, make any necessary changes to this program. Then, MERGE the next program. For example: MERGE "PROG2":

```
10  "B"  REM This is MERGED program
20  INPUT "Enter depreciation: ";D
30  INPUT "Number of Years:  ";Y
40  ... (etc.)
```

Now you may make changes to the above program since it was the last merged portion. If you need to make further changes to the first program then follow this procedure:

1. CSAVE what you have done to this point; use a new name when saving the tape, i.e., "PROG3" (as an example).
2. CLOAD the program saved in step 1. back into memory.
3. Now make any desired modifications or changes. However, keep this in mind: changes can be made *only* to the *first occurrence* of any duplicated line numbers. Thus, if program line 10 appears in the first program as well as the second, changes will only affect the first occurrence of line 10. If you attempt to edit the second occurrence of line 10, the change will be erroneously reflected in the first program portion only.

Adding Program Lines

Additional lines may be added at the end of the existing programs *only if they have line numbers greater than those previously used.* Note that additional program lines may not necessarily appear at the "bottom" of the listing. Since modules must be linked via labels (see below), this should not be of concern.

Linking Merged Modules (Programs) Together

Since the processor executes your program lines in logical sequence, it will stop when it encounters a break in the sequence in line numbering, i.e., if line numbers 10, 20, 30 are followed by duplicate line numbers in a second module, the processor will cease execution after line 30 in the first module.

To *link* your various modules together, the following techniques are valid: GOTO "B", GOSUB "B", IF . . . THEN "B" (where "B" is used for example only; you may use any label except reserved words or letters which appear in row number two on the keyboard, i.e., Q through P).

CREDIT WHERE CREDIT IS DUE

The Electronic Filter Program published on page 7 in Issue 15 of *PCN* was submitted by: *David L. Shuman, 1989 Altamont Place, San Diego, CA 92139.* We regret inadvertently failing to acknowledge that fact at the time of publication.

PRINTER'S HELPER

The accompanying listing presents a program of particular use to printers, publishers, art studios, typesetters and anyone who specifies type or sizes artwork and photographs.

The program has two sections which may be selected using the DEFine mode. SHIFT/A calls up the typesetting portion, SHIFT/B the art sizing routine.

Selecting SHIFT/A starts the typesetting routine by asking the user to input point size, leading and line length of the proposed typeset copy. A typical input here might be 10 point type on 11 points leading with a 22 pica line length.

Line 80 of the program asks if the user wishes to "compensate." If the response is affirmative here, the program reduces the number of characters computed to fit on each typeset line by five. This factor, five, is the average number of characters that is typically lost when setting justified or ragged-right copy. If "compensation" is not requested, then the program emulates most manual typefitting routines.

Lines 130 – 200 provide the characters-per-pica (CPP) count for some standard text typefaces: English Times, Helios, Helios Condensed, Helios Light, Clearface, Korina, Oracle and Souvenir.

Be advised that CPP values for typefaces can vary among phototypesetter manufacturers and typesetters. To customize the program for different CPP values, use the reciprocal of the 10 point value for the variable C. Thus, if 10 point Helios has a CPP of 2.43, set C equal to 1/(2.43 * 0.1). Incidently, the values provided in the listing are for Compugraphic Corporation typefaces.

If the typeface is not one of those provided in the listing, use the MAN (manual) selection and input a CPP value to two decimal places, such as 2.43.

The program outputs the calculated number of typeset lines, copy depth in points and inches.

Line 280 causes the program to recycle without the need to re-specify parameters that are not being altered.

Sizing Artwork

SHIFT/B is a handy routine for sizing artwork. The program asks for original dimensions in inches and tenths. New width, height or percent reduction or enlargement can then be selected.

This program supplied by: *Herb Griffiths, 129-1/2 Columbus Ave., Sandusky, OH 44870.*

Program *Printer's Helper*

```
10  GRAPHIC
20  "A" CLEAR
30  INPUT "CHRS PER MMS LINE? ";B
40  INPUT "# OF MMS LINES? ";A
50  INPUT "POINT SIZE? ";S
60  INPUT "LEADING? ";L
70  INPUT "LINE LNGTH/PICAS? ";D
80  INPUT "COMPENSATE (Y/N)? ";Z$
90  INPUT "TYPE STYLE? ";Q$
100 IF Z$="Y" LET H=5
110 IF Z$="N" LET H=0
120 IF Q$="MAN" GOTO 290
130 IF Q$="ENG" LET C=1/(S*.03731)
140 IF Q$="HEL" LET C=1/(S*.04115)
150 IF Q$="HELC" LET C=1/(S*.03344)
160 IF Q$="HELL" LET C=1/(S*.03802)
170 IF Q$="CLE" LET C=1/(S*.03322)
180 IF Q$="KOR" LET C=1/(S*.04000)
190 IF Q$="ORA" LET C=1/(S*.03861)
200 IF Q$="SOU" LET C=1/(S*.03571)
210 IF C=0 PRINT "BAD ENTRY":GOTO 90
220 E=(A*B)/(C*D-H)+.9:E=INT E
230 F=L*E:G=F/72
240 PRINT "# TYPESET LINES ";E
250 PRINT "COPY DEPTH/PNTS ";F
260 PRINT "COPY DEPTH/INCHES ";USING "####.##";G
270 USING
280 GOTO 30
290 INPUT "CHRS. PER PICA? ";C:GOTO 220
305 "B" CLEAR
310 INPUT "ORIGINAL WIDTH? ";J
320 INPUT "ORIGINAL HEIGHT? ";K
325 L=0:M=0:N=0
330 INPUT "WIDTH WANTED? ";L
335 IF L LET Q=L/J:P=Q*K:O=L:Q=Q*100:GOTO 360
340 INPUT "HEIGHT WANTED? ";M
345 IF M LET Q=M/K:O=Q*J:P=M:Q=Q*100:GOTO 360
350 INPUT "% WANTED? ";N
355 N=N*.01:O=J*N:P=K*N:Q=N*100
360 USING "####.##"
370 PRINT "W=";O;"   H=";P
380 PRINT "PERCENT= ";Q
390 GOTO 325
400 END
```

USING THE PANASONIC HHC FOR TELECOMPUTING

"Telecomputing" is a term for which Friends Amis, Inc., claims a trademark. Essentially, it stands for a method of communicating with a computer by remote means. Recently, I checked out this method of remote computer communications using the Panasonic HHC equipped with the model RL-P4001 acoustic modem (controlled by a Telecomputing 2 ROM) and the RL-P2001 TV adaptor. While the TV adaptor is not necessary for "Telecomputing," it does make the process somewhat easier during lengthy sessions.

The RL-P4001 modem is controlled from the HHC's keyboard. You activate its send and receive capabilities by selecting the appropriate items from the I/O (input/output) menu which is always available to the user. Next you use the system's primary menu to indicate that you want to use the "Telecomputing 2" system package. This activates software stored in a ROM in the modem unit. This software takes care of all communications protocols.

The Telecomputing 2 software permits you to configure all the necessary parameters for communicating with a remote computer over telephone lines. For instance, you can specify the baud rate, number of data bits, start anc stop bits, full or half-duplex modes, etc. Default values are set to those most commonly used by major remote computing services such as The SOURCE. In the tests I made with several local "bulletin board" services, I never had to change the basic configuration parameters. However, I did exercise the software to ascertain that it was easy to make such customized alterations if necessary.

To begin talking with a remote computer site, you just dial the telephone number of the "host" system and wait for the steady audio tone produced by the remote modem. Then you mount the telephone handset into the soft rubber cups of the modem. Next you select the "connect" option from the Telecomputing 2 menu. In a few seconds you see the beginnings of the "log-in" procedure come up on the LCD of the HHC and/or the TV monitor.

I tried the system both with and without the use of a TV monitor. Using a TV definitely makes things easier if you are having a lengthy Telecomputing session. But, without a large-screen display, the Telecomputing 2 system has features that make the use of the HHC practical. For example, you can select a mode called SLOW. When in this mode the HHC displays communications from the remote computer one line at a time. That is, one line of the HHC's display at a time. Whenever you are ready for the next line of information, you just press the NEXT key. That way, the information doesn't go zipping by you at 30 characters per second – unless you want it to. Press another button and the unit flips to the FAST mode whereby information is displayed as fast as it is received. Once you are familiar with a particular system you can use this mode to "log-in" or scan directories, etc.

I did find use of the HHC display alone somewhat tiring after extended use. The short lines, the effects of wordwrap, and constant pressing of the "next line" button soon seems to take the "fun" out of communicating using this method. However, there is no doubt that this is still a very viable method of getting a quick update or inputting data to a system. It also seems to me that familiarity with a particular system – getting to know what responses are expected to various prompts – undoubtably eases the operators burden when using the HHC in a standalone fashion.

Adding the TV monitor to the system makes even extended sessions quite enjoyable. The video display produced by the RL-P2001 TV adaptor produces one of the crispest outputs I have seen on a color TV set. The unit produces a solid green background color and then displays white characters against this background. Sixteen lines with up to 32 characters-per-line (and wordwrap) provide sufficient information on the screen at all times and considerably lessens the mental strain of remembering what you are doing as you "Telecompute." Connection of the TV adaptor to the TV is accomplished readily through one cable. The TV interface is activated by a simple menu selection on the HHC.

The bottom line is that a highly portable system capable of easily communicating with a remote computer (while having its own computing capabilities) is here – and it works. Furthermore, it is easy to use. I like it. *– Nat Wadsworth, Editor*

FROM THE WATCH POCKET

Where is Radio Shack's PC-2? As this issue went to press I had yet to find a R.S. store that would admit they had units for sale. Yet several readers say they have units in their hands? Chances are good they will be in the stores nationwide by early July.

There is a new PC program publisher on the scene and the initial releases indicate they are doing a good job. We have several of their new books containing numerous programs out for review. Write to: **Micro Text Publications, Inc.**, *One Lincoln Plaza, Suite 27C, New York, NY 10023*, or phone (212) 877-8539 and ask for a catalog describing their publications: *Pocket Magic, Pocket Computer Primer* and *Science & Engineering Sourcebook.* You can also get cassettes containing all the programs (about 25 per book) at a most reasonable price when you purchase the relevant title.

Prices on PCs and HHCs are beginning to drop rapidly (as I told you would occur in my December, 1981, summary). For instance, Radio Shack's original TRS-80 Pocket Computer, introduced at a price of $249.00 is now down to $149.00. This price is likely to drop still further (possibly below $100) with the introduction of the PC-2. Panasonic's 4K version of its HHC now has a list price of $418.00, down from $600.00 just six months ago. Rumors abound that Casio will soon have something similar to the FX-702P in capability at less than $100.00. And how about that move by Sinclair? Dropped the price of the ZX81 (assembled) from $149.95 to $99.95, the 16K RAM add-on was cut in half from $99.95 to $49.95, and the ZX81 kit came down from $99.95 to $79.95! A fully assembled and tested 16K, highly portable computer can now be attained for less than $150.00. Alas, the keyboard on that unit drives me bananas.

And where is HP's entry? I continue to hear talk of a powerful, multi-language capability HHC with 32K of memory (and more?) coming from that highly talented firm. End of the summer?

If you want to navigate by the stars in your yacht, be sure to read the article on using your TRS-80 PC for this purpose in the June, 1982, issue of **Microcomputing**, starting on page 112.

Cass Lewart has a nice article in the June issue of **Popular Electronics** that describes how to link up a PC-1 with an external printer.

I have heard that if you have a 2K Panasonic/Quasar HHC, you can expand it to 4K by buying a HM6116 memory IC at a cost of about $9.00 and plugging it into the back of the unit yourself.

The next issue of *PCN* comes out in September. Hope everyone has a pleasant summer. — *Nat Wadsworth, Editor*

SUBSCRIPTION INFORMATION

Subscriptions are sold on a calendar year basis (January – December). There are ten issues per year. You get all issues published for the calendar year to which you subscribe.

☐ 1982 Regular Subscriber (Issues 11 – 20). $30.00 in U.S. (U.S. $36.00 to Canada. U.S. $45.00 elsewhere.)

☐ 1982/83 Subscriber (Issues 11 – 30). $60.00 in U.S. (U.S. $72.00 to Canada. U.S. $90.00 elsewhere.)

• • • •

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER. Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: ____ Zip: _____

MC/VISA # _____ Expires: _____

Signature: _____

P.O. Box 232, Seymour, CT 06483

AN INDEX FOR THE SHARP PC-1500 INSTRUCTION MANUAL

Howard R. Battan, 35432 44th Avenue South, Auburn, WA 98002, got fed up with the lack of an index in the PC-1500's manual – so he created his own. Then, he sent in a copy to share with *PCN* readers!

& 98
abbreviations . . . 40, Apndx
ABS. 66
ACS. 65
AND 98
AREAD 133
arrays. 67
ARUN 134
ASC. 72
ASN 65
ATN 65
BEEP. 84
BEEP OFF. 85
BEEP ON 85
CHAIN. 113
CHR$. 73
clear key 13
CLOAD 109
CLOAD?. 109
CLS. 88
COLOR 121
concatenation (+). 70
CONT 100
COS. 65
CSAVE. 109
CSIZE 120
CURSOR. 85
DATA 55
DEG 66
DEGREE. 64
DIM. 67
DMS 66
editing 19
END 43
EXP. 65
FOR-TO-STEP 51
GCURSOR. 88
GLCURSOR. 124
GOSUB. 58
GOTO 46
GPRINT 91
GRAD 64
GRAPH 118
hexadecimal (&). 98
IF-THEN 44
INKEY$ 74
INPUT 35
INPUT# 110
INT. 66
LEFT$. 75
LEN 75
LET. 29
LF 121
LINE. 125
line numbers. 27
LIST 43
LLIST 118
LN 65
LOCK 100
LOG 65
logic operations 18
lower case 12
LPRINT 122
MERGE 111
MID$. 76
mode key 15
multiple statements. 41
NEW 29
NEXT 51
NOT 99
ON ERROR GOTO. 84
ON GOSUB 83
ON GOTO 83
OR 99
PAUSE. 34
Pi (π) 64
POINT 94
power on/off 63
PRINT 29
PRINT# 110
printer/cassette intfce . . 101
RADIAN. 64
RANDOM 78
range of numbers 62
READ 55
REM 57
RESTORE. 55
RETURN 58
RIGHT$ 77
RLINE 127
remote off 114
remote on 114
RND 77
ROTATE. 120
RUN 44
scientific notation. 60
SGN 67
SIN 65
SORGN 124
square root. 62
STATUS 79
STOP. 100
STR$. 78
TAB 124
TAN 65
TEST. 116
TEXT. 118
THEN 44
TIME. 79
TROFF. 96
TRON 96
two tape recorders 114
UNLOCK 100
USING 80
VAL 79
variables 23, 135
variables, string 23, 70
WAIT. 54
@( ). 135